Jacob Sunia

Prorgramming Assignment No. 5

CECS 229

In [2]:
```
import numpy as np

kobe = [[18, 7.6], [19, 15.4], [20, 19.9], [21, 22.5], [22, 28.5], [23, 25.2],
        [24, 30], [25, 24], [26, 27.6] , [27, 35.4], [28, 31.6], [29, 28.3],
        [30, 26.8], [31, 27], [32, 25.3], [33, 27.9], [34, 27.4], [35,13.8],
        [36, 22.3], [37, 17.6]]

kobe_np = np.array(kobe)
print(kobe_np)
print(type(kobe_np))
print(kobe_np.shape)
```

```
[[18.   7.6]
 [19.  15.4]
 [20.  19.9]
 [21.  22.5]
 [22.  28.5]
 [23.  25.2]
 [24.  30. ]
 [25.  24. ]
 [26.  27.6]
 [27.  35.4]
 [28.  31.6]
 [29.  28.3]
 [30.  26.8]
 [31.  27. ]
 [32.  25.3]
 [33.  27.9]
 [34.  27.4]
 [35.  13.8]
 [36.  22.3]
 [37.  17.6]]
<class 'numpy.ndarray'>
(20, 2)
```

In [41]:
```
#Part 3a Transpose
def transpose(kobe_np):
    kobe_transpose = np.transpose(kobe_np)
    return(kobe_transpose)


kobe_transpose = transpose(kobe_np)
print("Kobe Transpose:")
print(kobe_transpose)
```

```
Kobe Transpose:
[[18.  19.  20.  21.  22.  23.  24.  25.  26.  27.  28.  29.  30.  31.
  32.  33.  34.  35.  36.  37. ]
 [ 7.6 15.4 19.9 22.5 28.5 25.2 30.  24.  27.6 35.4 31.6 28.3 26.8 27.
  25.3 27.9 27.4 13.8 22.3 17.6]]
```

In [42]:
```
#Part 3b Make vetor ones
def vector_ones(kobe_np):
    kobe_rows = np.shape(kobe_np)[0]
    kobe_ones = np.ones(kobe_rows)
    return(kobe_ones)

ones = vector_ones(kobe_np)
print("Vector of Ones:")
print(ones)
```

```
Vector of Ones:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

In [45]:
```
#Part 3c Accessing Specific Elements in a numpy array
def specific_ele(kobe_np):
    A = kobe_np[:,0]
    y = kobe_np[:,1]
    return(A,y)
A,y = specific_ele(kobe_np)

print("Variable A:")
print(A)
print("\nVariable y:")
print(y)
```

```
Variable A:
[18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35.
 36. 37.]

Variable y:
[ 7.6 15.4 19.9 22.5 28.5 25.2 30.  24.  27.6 35.4 31.6 28.3 26.8 27.
 25.3 27.9 27.4 13.8 22.3 17.6]
```

In [46]:
```
#Part 3d: Concatenate two numpy arrays
def concatenate(col1,col2):
    x = np.column_stack((col1,col2))
    return(x)

x = concatenate(A,ones)
print("Concatenation of A and ones:")
print(x)
```

```
Concatenation of A and ones:
[[18.  1.]
 [19.  1.]
 [20.  1.]
 [21.  1.]
 [22.  1.]
 [23.  1.]
 [24.  1.]
 [25.  1.]
 [26.  1.]
 [27.  1.]
 [28.  1.]
 [29.  1.]
 [30.  1.]
 [31.  1.]
 [32.  1.]
 [33.  1.]
 [34.  1.]
 [35.  1.]
 [36.  1.]
 [37.  1.]]
```

In [49]:
```
#Part 3e: Matrix Multiplication
def matrix_mul(mat1, mat2):
    x_prod = np.matmul(mat1,mat2)
    return(x_prod)

x_transpose = transpose(x)
x_prod = matrix_mul(x_transpose,x)

print("Product of x_transpose and x:")
print(x_prod)
```

```
Product of x_transpose and x:
[[15790.   550.]
 [  550.    20.]]
```

In [50]:
```
#Part 4: On your Own
x_prod_inv = np.linalg.inv(x_prod)
y_prod = matrix_mul(x_transpose,y)
theta = matrix_mul(x_prod_inv,y_prod)

print("x_prod_inv:")
print(x_prod_inv)
print("\nTheta[0]:")
print(theta[0])
print("\nTheta[1]:")
print(theta[1])
```

```
x_prod_inv:
[[ 0.00150376 -0.04135338]
 [-0.04135338  1.18721805]]

Theta[0]:
0.199323308270678

Theta[1]:
18.72360902255639
```

In [ ]: